

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 07-219789

(43)Date of publication of application : 18.08.1995

(51)Int.Cl.

G06F 9/46

(21)Application number : 06-262140

(71)Applicant : INTERNATL BUSINESS MACH CORP <IBM>

(22)Date of filing : 26.10.1994

(72)Inventor : AULT DONALD F
ERNEST SCOTT BENDER
JOHN KEVIN FRANKS
JOHN ARTHUR HELMBOLD

(30)Priority

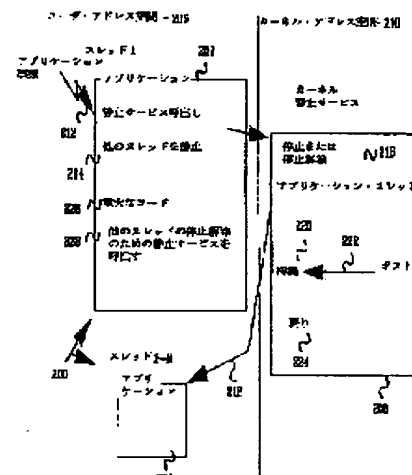
Priority number : 94 187675 Priority date : 27.01.1994 Priority country : US

(54) METHOD FOR PROCESSING EXTERNAL EVENT IN PLURAL THREAD SYSTEMS

(57)Abstract:

PURPOSE: To provide a method for processing an external event through a 1st thread by adjusting the stop of additional thread of plural thread applications.

CONSTITUTION: In response to the detection of the external event, the 1st sled transmits the event of stop for stopping that execution to the other additional thread of the application. Each thread does not hold severe resources. Therefore, when it is judged that it is safe to stop, each thread stops. The 1st sled is restarted before the stop of the final thread, and it is possible to perform an important operation freely without being interfered from the other additional threads. When the stop is an interruptive type, the interrupted execution of the other additional thread is restarted as soon as the important operation is completed, and the application recovers its ordinary operation.



LEGAL STATUS

[Date of request for examination] 26.10.1994

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number] 2856681

[Date of registration] 27.11.1998

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

(11)特許出願公開番号

特開平7-219789

(43)公開日 平成7年(1995)8月18日

技術表示箇所

3 4 0 B 7629-5B

審査請求 有 請求項の数14 O L (全 12 頁)

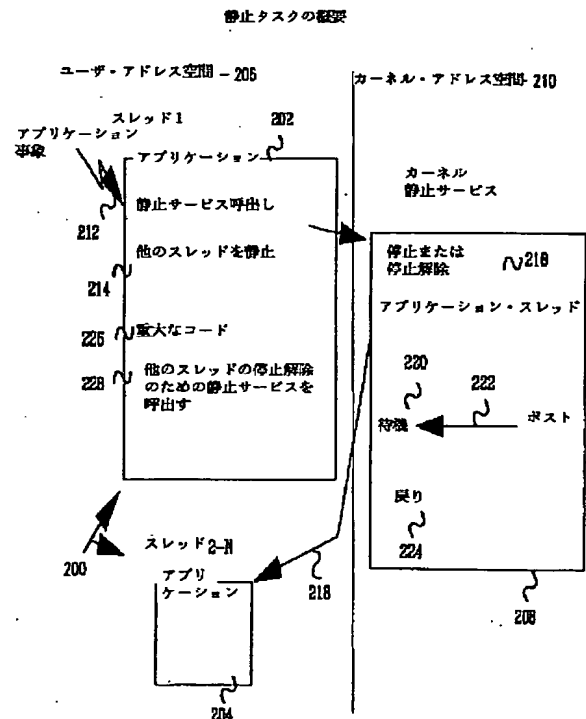
最終頁に続く

(54) 【発明の名称】 複数スレッド・システムにおける外部事象を処理する方法

(57) 【要約】

【目的】複数スレッド・アプリケーションの付加スレッドの静止を調整して、第1のスレッドによる外部事象の処理を行う方法を開示する。

【構成】外部事象の検出に応答して、第1のスレッドは、アプリケーションのその他の付加スレッドへ、その実行を中断するため静止事象を送信する。各スレッドは重大な資源を保持せず、従って静止することが安全であると判断される場合に、静止する。最後のスレッドが静止する前に第1のスレッドは再開され、他の付加スレッドからの干渉なしに重大な動作を自由に行うことができる。静止が中断タイプであれば、重大な動作の完了次第、中断した他の付加スレッドの実行を再開し、アプリケーションはその正常動作に戻る。



【特許請求の範囲】

【請求項1】第1のスレッドと第2のスレッドが共通アドレス空間で並列的に実行されるコンピュータ・システムにおいて、前記第1のスレッドによって外部事象を処理するための方法であって、
前記第2のスレッドを静止させるため、前記第1のスレッドから前記第2のスレッドへ静止事象を送信するステップと、
前記第2のスレッドへ送信された静止事象に応答して前記第2のスレッドが動作を静止するまで、前記第1のスレッドの実行を中断するステップと、
前記第2のスレッドへ送信された静止事象に応答して前記第2のスレッドが動作を静止した時、前記事象を処理するため前記第1のスレッドの実行を再開するステップと、
を含む方法。

【請求項2】前記第2のスレッドは、前記アドレス空間において前記第1のスレッドと並列に実行される複数の付加スレッドの1つであり、
前記第1のスレッドから前記複数の付加スレッドの各々に前記静止事象が送信される、
請求項1記載の方法。

【請求項3】前記複数の付加スレッドが当該スレッドへ送信された静止事象に応答して動作を静止するまで、前記第1のスレッドの実行が中断される、
請求項2記載の方法。

【請求項4】静止すべき前記複数の付加スレッドの最後のスレッドが、前記第1のスレッドの実行を再開する、
請求項3記載の方法。

【請求項5】前記第1のスレッドによる前記外部事象の検出に応答して、前記静止事象が前記第1のスレッドから前記第2のスレッドへ送信される、
請求項1記載の方法。

【請求項6】前記静止事象が、前記第2のスレッドを終了させるための終了事象である、
請求項1記載の方法。

【請求項7】前記静止事象が、前記第2のスレッドを中断させるための中断事象である、
請求項1記載の方法。

【請求項8】静止事象を前記第1のスレッドから前記第2のスレッドへ送信する前記ステップが、前記第2のスレッドの実行に割り込み、静止出口ルーチンに制御を与えるステップを含む、
請求項1記載の方法。

【請求項9】前記静止出口ルーチンが、前記第2のスレッドがシステム環境に影響を及ぼす程重大な資源を保持しているか否かを判断し、前記第2のスレッドが重大な資源を保持していないと判断する場合、前記第2のスレッドを静止させる、
請求項8記載の方法。

【請求項10】前記第2のスレッドがシステム環境に影響を及ぼす程重大な資源を保持しているか否かを判断するステップと、
前記第2のスレッドが重大な資源を保持していないと判断する場合、前記第2のスレッドを静止させるステップと、
を更に含む請求項1記載の方法。

【請求項11】前記第2のスレッドを静止させる前に前記第2のスレッドによって保持される重大な資源を解放するステップを更に含む、
請求項10記載の方法。

【請求項12】第1のスレッドと第2のスレッドが共通アドレス空間で並列的に実行されるコンピュータ・システムにおいて、前記第1のスレッドによって外部事象を処理するための方法であって、
前記第2のスレッドを中断させるため、前記第1のスレッドから前記第2のスレッドへ中断事象を送信するステップと、
前記第2のスレッドへ送信された中断事象に応答して前記第2のスレッドが動作を中断するまで、前記第1のスレッドの実行を中断するステップと、
前記第2のスレッドへ送信された中断事象に応答して前記第2のスレッドが動作を中断した時、前記事象を処理するため前記第1のスレッドの実行を再開するステップと、
前記第1のスレッドによる前記事象の処理の後、前記第2のスレッドの実行を再開するステップと、
を含む方法。

【請求項13】複数の付加スレッドが前記アドレス空間において前記第1のスレッドと並列に実行され、
前記第1のスレッドから前記複数の付加スレッドの各々に前記中断事象が送信される、
請求項12記載の方法。

【請求項14】前記複数の付加スレッドが当該スレッドへ送信された中断事象に応答して動作を中断するまで、前記第1のスレッドの実行が中断される、
請求項13記載の方法。

【発明の詳細な説明】**【0001】**

【産業上の利用分野】本発明は、複数スレッドを実行するプロセスにおける種々のスレッドの静止（すなわち、終了または中断）を調整する方法に関するものである。

【0002】

【従来の技術】コンピュータのオペレーティング・システム、すなわち、ユーザ・アプリケーションとハードウェアの間のインターフェースをとり、コンピュータ・システムの基本的監視機能を遂行するソフトウェアは当業者によく知られている。多くの現在のオペレーティング・システムは、1つのプロセスまたはアプリケーションの中で複数のスレッドを許容する。複数スレッド・アプ

リケーションとは、その仕事を実行するため1つ以上のスレッドの制御を使用するプログラムと定義される。

(本明細書において、用語「プロセス」と「アプリケーション」は、同義語として使われ、両者とも、共通アドレス空間を共有する1つ以上のスレッドを参照する。) A. S. Tanenbaum著Modern Operating Systems, (1992)において、いくつかの現在のオペレーティング・システムが一般的に説明され、特にその507頁から523頁においてスレッドが記述されている。

【0003】複数スレッド・アプリケーションをサポートするオペレーティング・システムの特定の例は、最近発表されたMVS拡張機能オープン・エディション (OpenEdition) を持つIBM MVS/ESA オペレーティング・システムである。MVS拡張機能OpenEditionを用いることによって、IBMシステム/390コンピュータとMVS/ESAオペレーティング・システムからなるハードウェア/ソフトウェア・プラットフォーム上で動作し、かつ、IEEE POSIX 1003.1、1003.2および1003.4ドラフト標準に準拠したアプリケーションをプログラムすることができる。(IBM、OpenEdition、MVS/ESAおよびSystem/390は、インターナショナル・ビジネス・マシーンス・コーポレーションの登録商標である。) MVS拡張機能OpenEditionの更に多くの情報は、本明細書で引用しているが、下記の刊行物で参照することができる: Ault著 "Fork Clone Address Space Implementation on MVS", IBM Technical Disclosure Bulletin, vol. 35, no. 6, pp. 363-67 (Nov. 1992); Ault著 "Interoperability Between MVS and POSIX Functions", IBM Technical Disclosure Bulletin, vol. 35, no. 6, pp. 383-88 (Nov. 1992); Ault他著 "Cross-Address Space Control Function", IBM Technical Disclosure Bulletin, vol. 36, no. 10, pp. 591-95 (Oct. 1993); Introducing OpenEdition MVS, IBM営業資料番号 GC233010-00 (1993); MVS/ESA Support for IEEE POSIX Standards: Technical Presentation Guide, IBM営業資料番号GG24-3867-0 (1993). 上述のように、MVS/ESAオペレーティング・システムのMVS拡張機能OpenEditionは、1つのプロセス内で複数のスレッドの使用を許容する。MVSの用語において、スレッドは、1つのタスクとみなすことができる。従って、複数スレッドは、1つのMVSアドレス空間内で複数のMVSタスクを使用することに等しい。

【0004】複数スレッド・アプリケーションは多くの状況において利点があるが、複数タスク (すなわち、複数スレッド) アドレス空間における適切なタスク制御が欠如しているため、終了、デバッグ作業およびダンプ処理の上で問題が起きる。このため、POSIX規格では、複数スレッドの1つが異常終了する場合当該プロセ

ス内のすべてのスレッドの終了が求められる。これは、MVSでは、ジョブ・ステップ・タスクの停止または該当タスクを停止するCallRTMを使用することによって実行される。しかし、OpenEditionのMVSスレッドをサポートするMVSタスクに上記のようなタイプの非同期停止を送信する場合、多くの問題に遭遇する。

【0005】発生する1つの問題は、プロセスのスレッドが突然ランダムに停止される場合、実行時ライブラリが共通のプロセス資源のクリーンアップを逐次処理することができないというものである。別の問題は、2つの命令の間で停止されつつある構成要素の多くは十分なエラー回復機能を持っていないというものである。場合によっては、これらの欠陥は、例えばファイルシステムの破壊といった破局的結果を招くことがある。停止誤り回復手順は改善できるが、この種の停止全体を回避することが望まれる。

【0006】制御された方法で複数スレッド・アプリケーションの残存スレッドを中断する必要性は、デバッグ作業において発生することがある。複数スレッド・アプリケーションをデバッグする際、デバッグを行うユーザは、実行するスレッドとある特定事象に対して中断すべきスレッドを選択でき、また、実行/中断状態を動的に変更することができることを必要とする。この中断プロセスは、また、アプリケーションの流れを変えるような種類のもの、あるいは、中断の時点で当該スレッドが重大なシステム管理資源を保持していることを許容するような種類のものではなならない。

【0007】複数スレッド・アプリケーションの残存スレッドを中断する必要性が発生する別の場合は、すべてのスレッドから得られる情報を用いて当該プロセスのダンプを行う時である。必要性は、上述したデバッグの状況と同様である。ダンプを要求するタスクは、呼び出し元タスクがダンプをとることを妨げるような重大なシステム資源をその他のアスクが保持していない場合そのようなその他のタスクのすべての実行を中断できなければならない。ダンプがとられたあと、ダンプを行うタスクは、他のタスクの実行を再開しなければならない。

【0008】

【発明が解決しようとする課題】このように、複数タスク・アドレス空間において適切なタスク制御が欠如する場合は、終了、デバッグおよびダンプの各局面において問題が発生する。複数スレッドが動作する環境において、予測可能でかつ非破壊的な形態でタスクの実行を終了あるいは中断させる仕組みが求められる。

【0009】

【課題を解決するための手段】上述の課題は、あるスレッドが起動された時点で当該アドレス空間に存在するすべての他のスレッド (すなわちタスク) に静止事象 (QUIESCE EVENT) を送信する新しい静止機能

を作成することによって解決される。静止機能を起動したスレッドは、すべての事象が実行され、目標スレッドが期待した状態に置かれるまで待機する。

【0010】これを達成するため、静止事象を受信次第オペレーティング・システムがその制御権を渡すべき静止機能出口(E X I T)ルーチンを、ユーザがオペレーティング・システムに通知するための登録機能が提供される。出口が指定されていない場合は、該事象をいつ処理するかはオペレーティング・システムが決定する。

【0011】事象の伝達は、目標スレッドの実行に割り込みを行うサービス要求ブロック／割り込み応答ブロック(SRB/IRB)というブロックの組み合わせによる。割り込みの時点で動作していた要求ブロック(RB)に関し静止事象を実行させることが安全であるかどうかを検査する種々のチェックがなされる。これらのチェックには、制御を静止機能出口に渡すことがシステム環境にとって受容可能であるかを確認することが含まれる。例えば、システム・サービスに割り込むことは望しくないからである。制御が静止機能出口に渡され、静止事象が実行されるべきであると判断されると、適切な処置がとられる。もしも静止事象が終了のためのものであったとすれば、出口ルーチンは当該スレッドの実行を終了する。静止事象がスレッドを中断するべきものであったならば、出口は、適切な中断サービスを提供する。

【0012】システムIRBまたはユーザの静止出口が静止事象を実行できないと判断すると、事象は未処理のままにされ、次のシステム・サービスの出口に再び伝達されるか、あるいは、それより早くユーザが静止事象を実行できる安全点に達したことを検知すればその時点でシステム・サービスの出口に再伝達される。

【0013】静止事象の伝達は、最後のスレッドが期待した状態に入るまですべてのスレッドに対して実施される。そのような最後のスレッドは、静止機能を起動した元のスレッドにポスト信号を送信し、該スレッドの待機を解く(すなわち、元のスレッドを再開する)。

【0014】この解決方法の利点は、静止状態がいつスレッドに効果を及ぼす状態に入るべきかを決定することが可能であるという点である。重大な資源を保持しているスレッドを無条件に停止させるという問題は回避される。コードの「不安定」または「重大な」部分で実行中のスレッドを非同期的に停止させることに起因する破壊的な結果もまた回避される。

【0015】

【実施例】図1は、本発明を組み入れているコンピュータ・システム100の概要であり、静止機能の実現のための各システム層の間の関係を示している。これらの層は、図1の上部から始まって、アプリケーション層102、言語サブルーチン／実行時ライブラリ(RTL)層104、オペレーティング・システムまたはカーネル層106、およびハードウェア層108である。

【0016】システム100の基部には、1つ以上の中央処理装置(CPU)、主記憶装置、および、磁気ディスク記憶装置、磁気テープ装置、端末、プリンタ等の入出力装置から構成されるハードウェア層108がある。これらの構成要素は当業者にとっては周知のものであるので、特に図示しない。

【0017】ハードウェア層108の上の層のカーネル層106は、CPU、記憶装置および入出力装置を管理する基本ハードウェア層を制御し、ハードウェア層における資源へのアプリケーション・コードのアクセスを可能にする呼び出し可能サービスを含むソフトウェアから構成される。(本発明に関連した静止サービスは上記呼び出し可能サービスの一部である。)カーネル層106は、IBMシステム/390コンピュータからなるハードウェア上で動く拡張機能OpenEditionを持つMVS/ESAオペレーティング・システムから構成される。しかし、本発明は、上記のようなハードウェア／ソフトウェア・プラットフォームに限定されるものではなく、他のプラットフォームを代替的に使うことができる。

【0018】言語サブルーチン／実行時ライブラリ(RTL)層104は、カーネル層106のすぐ上部に位置する。層104は、多くのアプリケーションによって使われる高級言語(HLL)をサポートするルーチンから構成される。層104は、基本的には、要求されたHLL機能を、その要求を実行するために適切なカーネル・サービスに変換する。

【0019】アプリケーション層102は、システム100の最上部の層であって、1つ以上のユーザ・アプリケーションを含む。アプリケーションは、カーネル106から直接に、あるいは、言語サブルーチン／実行時ライブラリ104を経由してHLLから、システム・サービスを要求することができる。本明細書の仕様の目的から、アプリケーション層102と言語サブルーチン／実行時ライブラリ104とは集合的に一括してアプリケーションとみなすことができる。

【0020】図2は、システム100において実施される本発明の静止機能の概略流れ図である。(停止、停止解除または終了を実行する)特定の静止機能に関するステップはこの流れ図にすべて含まれてはいないが、すべての静止機能はこの一般的流れ図に従う。

【0021】図2は、図1のシステム100のアプリケーション層102のアプリケーション200を示している。該アプリケーション200は、第1のスレッド202(スレッド1)と1つ以上のその他の付加スレッド204(スレッド2からN)を含む。スレッド202、204は、多くの独立プロセス属性を持つ。例えば、それ自身のプログラム・カウンタと状態(すなわち実行中、待機または実行阻止中等の状態)をそれぞれのスレッドが持つが、それらスレッドが共通のユーザ・アドレス

空間を共有するという点で独立プロセスとは相違する。

(アプリケーション200は、実際、同一アドレス空間を共有するスレッド202、204のセットとして定義することができる。)スレッド202は、必ずしも作成されるべきアプリケーション200の最初のスレッドではない;それが静止サービスを定義して起動するが故に、他のスレッド204と単に区別するために第1のスレッドと呼ぶにすぎない。

【0022】図2にはまた、システム100のカーネル層106に配置された本発明の静止サービス208が示されている。静止サービス208は、ユーザ・アドレス空間とは別のカーネル・アドレス空間210に位置し、以下のような独自のサービス(または機能)を提供する:

1. quiesce_exit_registration (出口登録) サービス306: スレッドが以下にリストされる静止機能の1つを要求した時に制御を受け取るべき静止出口ルーチンのアドレスを初期スレッドが提供することを可能にする。静止出口は、登録した初期スレッドと関係づけられるスレッドで発生する静止事象の結果として制御を受け取るアプリケーション・レベル・ルーチンである。この出口は、伝達された静止事象を実行させる責任がある。

2. quiesce_freeze (停止) サービス404: すべての他のスレッドに中断事象を送信し、ユーザ定義静止出口ルーチンに制御を渡す。この事象に関し、静止出口ルーチンは、重大な資源がスレッド204によって保持されているか判定する。重大な資源が保持されていなければ、静止出口ルーチンは、quiesce-freeze-selfサービスを呼び出す。重大な資源が保持されている場合、重大な資源が開放されるまで、静止事象の処理は遅延される。

3. quiesce_freeze_self (自己停止) サービス722: 起動元スレッドを中断する。このオプションは、中断事象の受信時に静止出口ルーチンによって使われる。

4. quiesce_event_put_back (処理遅延) サービス734: 静止事象の処理を後の時点まで遅延させる。このオプションは、現在の実行環境の事情のために処理できない中断事象の受信時に静止出口ルーチンによって使われる。

5. quiesce_unfreeze (停止解除) サービス806: 停止されたすべてのスレッドを再開する。

6. quiesce_term (終了) サービス904: すべての他のスレッドに終了事象を送信し、ユーザ定義静止出口がquiesce_exit_registrationサービスを用いて指定されている場合ユーザ定義静止出口に制御を渡す。この事象に関し、静止出口は、重大な資源がスレッド204によって保持されているか判断する。重大な資源が保持されていなければ、静止出口は、スレッドを終了させたためpthread_exitサービスを呼び出す。重大な資源が保持されている場合、重大な資源が開放されるまで、静止事象の処

理は遅延される。

7. quiesce_force (バイパス) サービス: すべての他のスレッドへ終了事象を送信し、ユーザ定義静止出口への呼び出しをバイパスさせる。

【0023】アプリケーション事象212がスレッド202上で検知される時、本発明の動作が始まる。アプリケーション200の他のスレッド204が実行中は、このアプリケーション事象212は処理されることができない。事象212は、実施例によっては、停止、プログラム・チェックまたはブレイク・ポイントであり得る。この時点で、アプリケーション200は、ダンプをとることができるように、あるいは、診断機能を実行することができるように、他のスレッド204の実行を中断することを必要とする場合がある。あるいはまた、アプリケーション200は、他のスレッド204が終了しなければならないことを当該他のスレッドに通知することを必要とする場合もある。

【0024】上述したように、デバッグ作業にも本発明を使うことができる。しかし、本発明はそのような使用に限定されることはなく、また、そのような使用の詳細は本発明の対象ではない。

【0025】本発明に従って、スレッド202が、アプリケーション200において他のスレッド204の静止を必要とする事象212を通知する時、アプリケーションの他のスレッドを静止するため、スレッド202は、カーネル静止サービス208を呼び出す(ステップ214)。静止サービス208は、静止通知218を他のスレッド204へ送信し(ステップ216)、静止通知が他のスレッドによって実行されるまで待機(すなわち中断)する(ステップ220)。静止通知218を受信次第、スレッド204は静止のタイプに基づいて該当する措置をとる。最後のアプリケーション・スレッド204は、静止通知を実行する時、カーネル静止サービス208で待機中のスレッド202にポスト信号を送る(すなわち、スレッド202を再開させる)。スレッド202へのポスト信号の送達とともに、静止サービス208は、アプリケーション200のスレッド202へ制御を戻す(ステップ224)。

【0026】スレッド202は、今や、他のスレッド204が動作中には処理することができなかった重大なコードを実行することができる(ステップ226)。この重大なコードの実行が完了し、当初の措置が他のスレッド204を停止させるべきものであったなら、スレッド202は、他のスレッドを停止解除するため静止サービス208を呼び出す(ステップ228)。

【0027】図3は、アプリケーション200が、静止事象を扱うアプリケーション定義ルーチン(図7から図9に記載)である静止出口をどのように登録するかを示す。以下詳細に説明する。

【0028】静止出口を登録するため、アプリケーショ

ン200の第1のスレッド302が、静止サービス208のquiesce_exit_registrationサービス306を呼び出し(ステップ304)、静止出口ルーチンのアドレス(quiesce_exit)を渡す。Quiesce-exit-registrationサービス306は、スレッド302を起動するためスレッド制御待ち行列エレメント(TCQE)308にアプリケーション静止出口アドレスを記憶する。図6に示されるように、TCQE308は、スレッド302とアプリケーション静止出口アドレス604と待機/ポストのための事象制御ブロック606とを識別するスレッド識別子602を含むメモリ上の定義された領域である。今や、スレッド302は静止事象に関して登録された。

【0029】スレッド302は、カーネル・アドレス空間におけるスレッド作成(pthread_create)サービス314を用いて、その他の付加スレッド312を作成することができる(ステップ310)。カーネル・スレッド作成サービス314は、適切な方法で実施されるが、その方法は当業者には周知であるので、本明細書には記載しない。作成する新しいスレッド312(ステップ316)の各々に関して、スレッド作成サービスは、スレッド302に関するTCQE308と同様のTCQE308を構築し(ステップ318)、固有のスレッド識別子602を割り当て、作成元スレッド302のTCQE308から新たに作成されるスレッド312のTCQE308へ静止出口アドレス604をコピーする。

【0030】かくして、アプリケーション中の全てのスレッド302、312が同一の静止出口アドレス604を持って登録された。図3で示されるように、各TCQE308は、適切な方法で(例えば、ポインタ、連続的メモリ・ロケーションまたはその他の方法で)次のスレッド312のためのTCQE308に連結し、以下に記述のようにチェーンまたはスレッド制御待ち行列(TCQ)320を形成し、このため、ある特定のアプリケーションに関するTEQE308は逐次走査できる。

【0031】図4は、図2の概略流れ図をより詳細にした本発明のプロセス流れ図である。始めに、アプリケーション200中のすべての他のスレッド204の停止を必要とする事象212が、アプリケーション・スレッド202上で発生する。事象212を検知する時、アプリケーション・スレッド202は、静止サービス208のquiesce_freezeサービス404を呼び出す(ステップ402)。quiesce_freezeサービス404は次に内部事象生成ルーチン408を呼び出す(ステップ406)。

【0032】アプリケーション200中の起動元スレッド202以外のスレッド204の各々に関して、事象生成ルーチン408は、静止出口インターフェース・ブロック(QEIB)412を先ず作成する。これは、TCQ320をサーチして、すべての今後のスレッド204を識別することによって達成される。図5を参照すると、QEIB412は、目標スレッドのアドレス502

と、静止事象のタイプ504(すなわち、停止か終了)と、割り込みプログラム状態語(PSW)およびレジスタ内容506とを記憶するためのアドレス・ロケーションを含むユーザ・アドレス空間において定義された領域である。事象生成ルーチン408は、初期的には、QEIB412に、目標スレッド・アドレス502と静止事象タイプ504を格納する。

【0033】次に、上記サーチによってTCQ320で見出されたスレッド204の各々に関し、事象生成ルーチン408は、サービス要求ブロック(SRB)416をスケジュールして(ステップ414)、スレッドへ停止要求割り込みを送信し、該スレッドを停止させる。SRB416の各々は、ユーザ・アドレス空間206において実行するためカーネルによってディスパッチされる1つの作業単位(unit of work)である。各SRB416は、目標スレッド204に対する割り込み要求ブロック(IRB)418を作成しスケジュールする。IRB418は、下記のように動作する。

【0034】すべての該当するスレッド204に対する割り込みをスケジュールした後、事象生成ルーチン408は、quiesce_freezeサービス404に制御を戻す(ステップ420)。Quiesce_freezeサービス404は、スレッド202に関するTCQE308に配置された事象制御ブロック606(図6)上で待機する(ステップ422)。この待機は、静止事象に応答する最後のスレッド204からのポスト信号によって、最後のスレッドが停止状態に入る直前に解かれる。

【0035】図7を参照して、上記割り込みメカニズムをより詳細に説明する。IRB418がアプリケーション200の制御を獲得すると、実行は、目標スレッド204上で(図の符号702の箇所)で停止する。IRB418は、アプリケーションのスレッド状態をチェックして、システム環境が静止事象を扱うことを受容できることを確認する。システム環境が受容可能である場合(ステップ704)、IRB418は、対応するQEIB412(図5)の部分506に、スレッド204に関するプログラム状態語(PSW)とレジスタ内容を保存する。プログラム状態語(PSW)とレジスタ内容は、スレッド204に関するシステム・ディスパッチャによって管理される対応タスク・レベル制御ブロック708から抽出される。PSWは、静止事象が処理された後再開されるべき割り込み箇所702をポイントする。次に、IRB418は、目標とされたスレッド204の再開PSWを以前に登録した静止出口712をポイントするように修正し(ステップ710)、レジスタ1をスレッド204のQEIB412をポイントするように修正する(ステップ714)。次にプロセスはIRB418から出て(ステップ716)、修正されたPSWとレジスタ1を用いて目標スレッド204で実行が再開される。

【0036】静止出口712は制御権を獲得しQEIB

412へのアクセスを有する。静止出口712はアプリケーション環境（すなわち、スレッドの実行状態）を検査し、プロセスを停頓させるような重大な資源をスレッド204が保持していないことを確認する。

【0037】システム環境が受容可能であれば（ステップ718）、静止出口712は、QEIB412の静止事象タイプ・フィールド504（図5）を調べ、どの事象タイプを処理するため呼び出されたか、また、それ自身を中断または終了させるべきかを判断する。あり得る2つの事象タイプは、スレッド204の中断を要求する事象とスレッドの終了を要求する事象とである。本例におけるように、事象がスレッド204の中断を要求するタイプである場合、静止出口712は、静止サービス208のquiesce_freeze_selfサービス722を起動する（ステップ720）。Quiesce_freeze_selfサービス722は、呼び出し元スレッド204がアプリケーション200中で静止状態になるべき最後のスレッドであることを調べる。そうならば（ステップ724）、Quiesce_freeze_selfサービス722は、quiesce_freezeサービス404に、従って、quiesce_freezeサービスを当初起動したスレッド202（図4）に、ポスト信号を送る（ステップ726）。Quiesce_freeze_selfサービス722は、スレッド204を待機状態に入らせるためシステム待機サービス呼び出すことによってスレッド204の実行を中断する（ステップ728）。

【0038】静止出口712は、アプリケーション環境が受容できないと判断する場合（ステップ730）、quiesce_event_put_backサービス734を起動して事象をカーネルへ戻す（ステップ732）。Quiesce_event_put_backサービス734は、静止事象をこの時点で取り扱うことができないこと、および、アプリケーション200が後刻事象の送達を要求するであろうこと、をカーネルに通知する。これは、静止事象がなお未処理状態にあることを示すため該当するTCQE308の事象制御ブロック606（図6）にマークをつけることによって行われる。環境から障害を取り除く責任は、アプリケーション200にある。アプリケーション・スレッド204が静止される準備状態になると、Quiesce_freeze_selfサービス722を呼び出すか、又は、代替的方法として、静止出口712を再起動するようカーネルに要求する。

【0039】上述のように、すべてのスレッド204がquiesce_freeze_selfサービス722を起動したとき、quiesce_freeze_selfサービス726は、静止停止サービス404（図4）にポスト信号を送り、制御は、その呼び手（アプリケーション200のスレッド202）に戻される。

【0040】上述のように、すべてのスレッド204がquiesce_freeze_selfサービス722を起動した時、quiesce_freeze_selfサービス722はquiesce_freezeサー

ビス404（図4）にポスト信号を送り、制御は、その呼び手（アプリケーション200のスレッド202）に戻される。図8を参照すると、今やすべての他のスレッド204が停止されているので、1つの実行中のスレッド202が、アプリケーション200中の他のスレッド204からの干渉なしに、要求される重大な作業を遂行することができる（ステップ802）。重大な作業が完了次第、スレッド202は、静止サービス208のquiesce_unfreezeサービス806を呼び出す（ステップ804）。quiesce_unfreezeサービス806は、pthread_create（図3）の間に構築されたスレッド制御待ち行列エレメント308のTCQ320を通して動き、quiesce_freeze_selfサービス722内で現在待機中のスレッド204の各々にポスト信号を送る（ステップ808）。ポスト信号発信次第、quiesce_freeze_selfサービスは、各停止スレッド204の制御を対応する静止出口712に戻し（ステップ810）、割り込み箇所702での実行を再開する（ステップ812）。この再開は、QEIB412に保存したPSWとレジスタ内容506を使用して行われる。次に、quiesce_unfreezeサービス806は、スレッド202に制御を復帰し（ステップ814）、アプリケーション処理を再開する（ステップ816）。

【0041】図9は、静止終了要求のプロセスの流れを示す。静止終了要求は、呼び出し元スレッド202を除き、アプリケーション中のスレッド204のすべてを終了させる機能を持つ。終了するスレッド204は、終了する前にアプリケーション上重大なコードを完了したり、あるいは、スレッド関連資源を問題の起さない状態にする機会を与えられる。

【0042】図9に示されるように、アプリケーション200中の他のスレッド204を終了させることを要求する事象212が通知され次第、スレッド202は、アプリケーションの他のスレッドの終了を要求するため、静止サービス904のquiesce_termサービス904を呼び出す（ステップ902）。quiesce_termサービス904は、TCQチェーン320を走査し、呼び出し元スレッド202に関するTCQEを除くすべてのチェーン上のTCQE308の各々に関するquiesce_term事象908を生成する。図9には示されていないが、このquiesce_term事象908は、第4図で示された方法と同様にサービス要求ブロック（SRBS）と割り込み要求ブロック（IRBS）とを使用してスレッド204へ伝達されることが好ましい。次に、quiesce_termサービス904は、他のスレッド204が終了させられるまで、待機する（ステップ910）。

【0043】quiesce_termサービス904によって生成された事象908をスレッド204が受信すると、アプリケーションの通常の流れは、上述の通り箇所702で割り込まれ、静止出口712に制御権が与えられる。静

止出口712は、停止に関し上述したケース(図7)と同様にアプリケーション環境をチェックする。アプリケーション環境が受容可能で、生成された静止事象が(QEIB412の静止事象タイプ・フィールド504によって標示されるが)終了のためのものであれば(ステップ912)、静止出口712は、スレッド・クリーンアップ・ルーチン(pthread_exit)922を起動する(ステップ914)。

【0044】Pthread_exitルーチン922は、終了するスレッド204に関係したシステム資源を解放する。当該終了スレッド204が、アプリケーション中で、生成されたquiesce_term事象を持った最後のスレッドであれば(ステップ916)、pthread_exitルーチン922は、quiesce_termサービス904で待機中のスレッド202にポスト信号を送る(ステップ918)。スレッド202が上記ポスト信号を受信しquiesce_termサービス904での待機を解いた後(ステップ918)、アプリケーション200に制御が戻され、受信した事象212に基づいて適切な措置が取られる(ステップ920)。

【0045】(図示されていないが)quiesce_forceサービスが、quiesce_term サービス904と同様に動作し、すべての他のスレッド204に終了事象を送信する。しかし、quiesce_forceサービスは、ユーザ定義静止出口の呼び出しをバイパスし、割り込み要求ブロック(IRB)からpthread_exitルーチン922が直接呼び出される。

【0046】まとめとして、本発明の構成に関して以下の事項を開示する。

(1) 第1のスレッドと第2のスレッドが共通アドレス空間で並列的に実行されるコンピュータ・システムにおいて、上記第1のスレッドによって外部事象を処理するための方法であって、上記第2のスレッドを静止させるため、上記第1のスレッドから上記第2のスレッドへ静止事象を送信するステップと、上記第2のスレッドへ送信された静止事象に応答して上記第2のスレッドが動作を静止するまで、上記第1のスレッドの実行を中断するステップと、上記第2のスレッドへ送信された静止事象に応答して上記第2のスレッドが動作を静止した時、上記事象を処理するため上記第1のスレッドの実行を再開するステップと、を含む方法。

(2) 上記第2のスレッドは、上記アドレス空間において上記第1のスレッドと並列に実行される複数の付加スレッドの1つであり、上記第1のスレッドから上記複数の付加スレッドの各々に上記静止事象が送信される、上記(1)記載の方法。

(3) 上記複数の付加スレッドが当該スレッドへ送信された静止事象に応答して動作を静止するまで、上記第1のスレッドの実行が中断される、上記(2)記載の方法。

(4) 静止すべき上記複数の付加スレッドの最後のスレ

ッドが、上記第1のスレッドの実行を再開する、上記(3)記載の方法。

(5) 上記第1のスレッドによる上記外部事象の検出に応答して、上記静止事象が上記第1のスレッドから上記第2のスレッドへ送信される、上記(1)記載の方法。

(6) 上記静止事象が、上記第2のスレッドを終了させるための終了事象である、上記(1)記載の方法。

(7) 上記静止事象が、上記第2のスレッドを中断させるための中断事象である、上記(1)記載の方法。

(8) 静止事象を上記第1のスレッドから上記第2のスレッドへ送信する上記ステップが、上記第2のスレッドの実行に割り込み、静止出口ルーチンに制御を与えるステップを含む、上記(1)記載の方法。

(9) 上記静止出口ルーチンが、上記第2のスレッドがシステム環境に影響を及ぼす程重大な資源を保持しているか否かを判断し、上記第2のスレッドが重大な資源を保持していないと判断する場合、上記第2のスレッドを静止させる、上記(8)記載の方法。

(10) 上記第2のスレッドがシステム環境に影響を及ぼす程重大な資源を保持しているか否かを判断するステップと、上記第2のスレッドが重大な資源を保持していないと判断する場合、上記第2のスレッドを静止させるステップと、を更に含む上記(1)記載の方法。

(11) 上記第2のスレッドを静止させる前に上記第2のスレッドによって保持される重大な資源を解放するステップを更に含む、上記(10)記載の方法。

(12) 第1のスレッドと第2のスレッドが共通アドレス空間で並列的に実行されるコンピュータ・システムにおいて、上記第1のスレッドによって外部事象を処理するための方法であって、上記第2のスレッドを中断させるため、上記第1のスレッドから上記第2のスレッドへ中断事象を送信するステップと、上記第2のスレッドへ送信された中断事象に応答して上記第2のスレッドが動作を中断するまで、上記第1のスレッドの実行を中断するステップと、上記第2のスレッドへ送信された中断事象に応答して上記第2のスレッドが動作を中断した時、上記事象を処理するため上記第1のスレッドの実行を再開するステップと、上記第1のスレッドによる上記事象の処理の後、上記第2のスレッドの実行を再開するステップと、を含む方法。

(13) 複数の付加スレッドが上記アドレス空間において上記第1のスレッドと並列に実行され、上記第1のスレッドから上記複数の付加スレッドの各々に上記中断事象が送信される、上記(12)記載の方法。

(14) 上記複数の付加スレッドが当該スレッドへ送信された中断事象に応答して動作を中断するまで、上記第1のスレッドの実行が中断される、上記(13)記載の方法。

【0047】

【発明の効果】本発明によって、静止状態がいつスレ

ドに効果を及ぼす状態に入るべきかを決定することが可能となる。また、重大な資源を保持しているスレッドを無条件に停止させるという問題が回避される。コードの「不安定」または「重大な」部分で実行中のスレッドを非同期的に停止させることに起因する破壊的な結果もまた回避される。

【図面の簡単な説明】

【図1】本発明を組み入れているコンピュータ・システムの概要を示す図である。

【図2】本発明に従って他のスレッドを静止（すなわち停止または終了）させるための一般的手続きを示す図である。

【図3】本発明に従って静止出口を初期登録するための手続きを示す図である。

【図4】本発明に従って他のスレッドを停止（すなわち、中断）する手続きの初期局面を示す図である。

【図5】図4で示される停止手続きの初期局面の間に作成される静止出口インターフェース・ブロック（QEIB）を示す図である。

【図6】図3で示される初期登録手続きの間に作成されるスレッド制御待ち行列エレメント（TCQE）を示す図である。

【図7】本発明に従って他のスレッドを停止させる手続きの最終局面を示す図である。

【図8】本発明に従って他のスレッドの停止を解く（すなわち、再開する）ための手続きを示す図である。

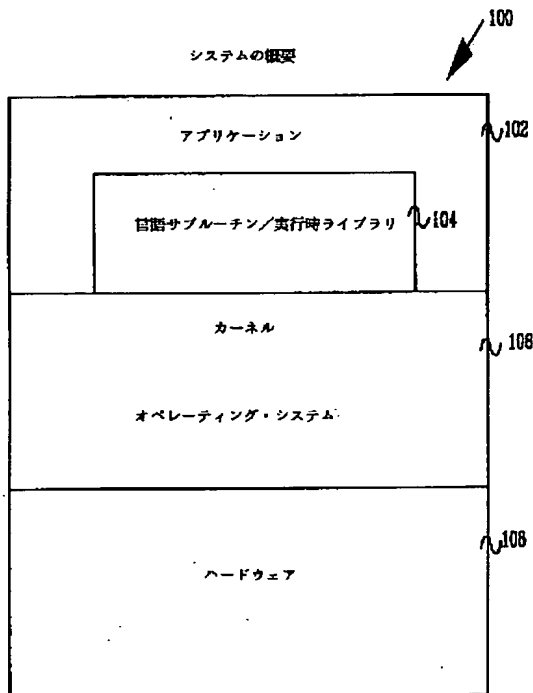
【図9】本発明に従って他のスレッドを終了させるための手続きを示す図である。

【符号の説明】

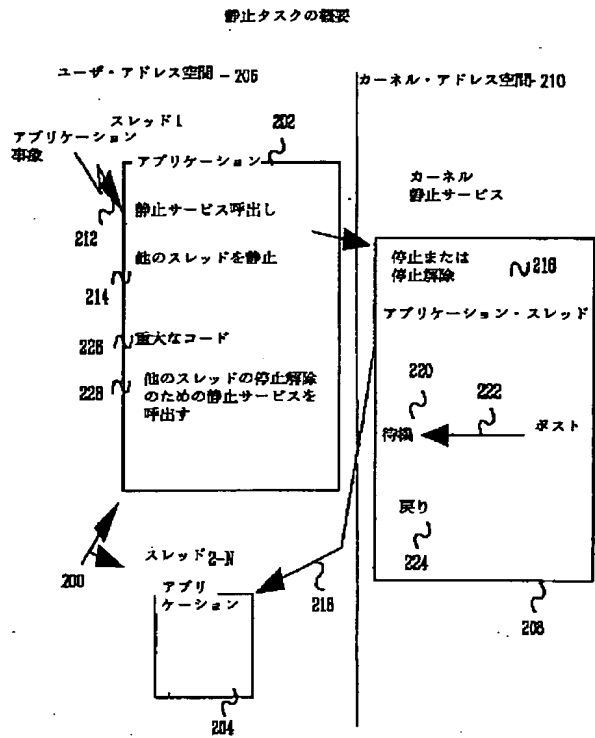
100	コンピュータ・システム
102	アプリケーション層
104	言語サブルーチン／実行ライブラリ層
106	オペレーティング・システムまたはカーネル層
108	ハードウェア層
200	アプリケーション
202、302	第1のスレッド
204、304、312	第2のスレッド
206	ユーザ・アドレス空間

208	静止サービス
212	アプリケーション事象
218	静止通知
306	quiesce_exit_registration
(出口登録) サービス	
308	スレッド制御待ち行列エレメント (TCQE)
314	pthread_create (スレッド作成) サービス
320	スレッド制御待ち行列 (TCQ)
404	quiesce_freeze (停止) サービス
408	内部事象生成ルーチン
412	静止出口インターフェース・ブロック (QEIB)
416	サービス要求ブロック (SRB)
418	割り込み要求ブロック (IRB)
502	目標スレッド・アドレス
504	静止事象タイプ
506	PSWとレジスタ内容
602	スレッド識別子
604	アプリケーション静止出口アドレス
606	事象制御ブロック
702	割り込み箇所
712	静止出口
722	quiesce_freeze_self (自己停止) サービス
734	quiesce_event_put_back (処理遅延) サービス
806	quiesce_unfreeze (停止解除) サービス
904	quiesce_term (終了) サービス
908	quiesce_term事象
922	スレッド・クリーンアップ・ルーチン (pthread_exit)

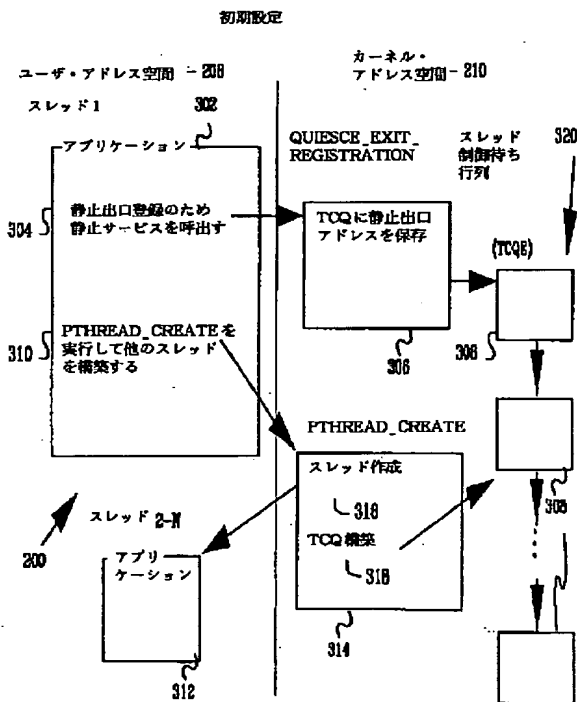
【図 1】



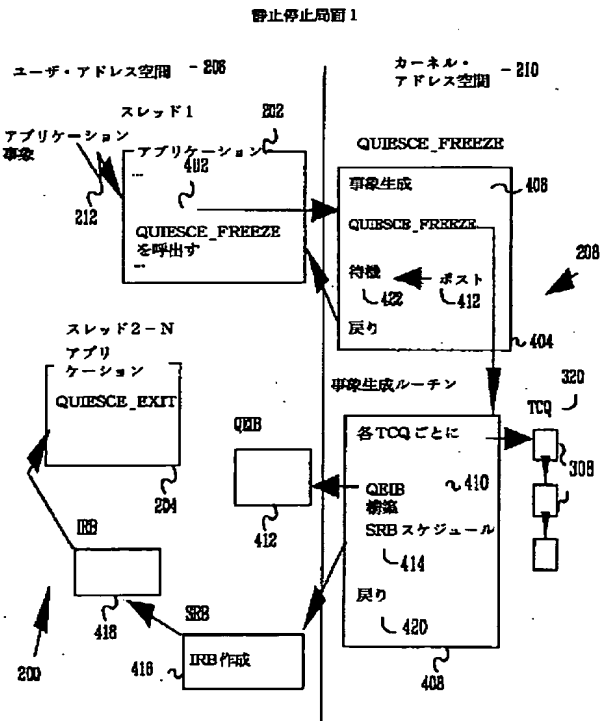
【図 2】



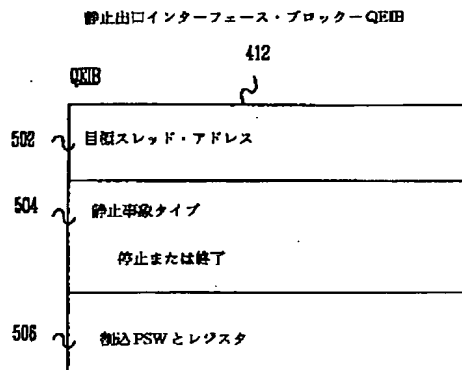
【図 3】



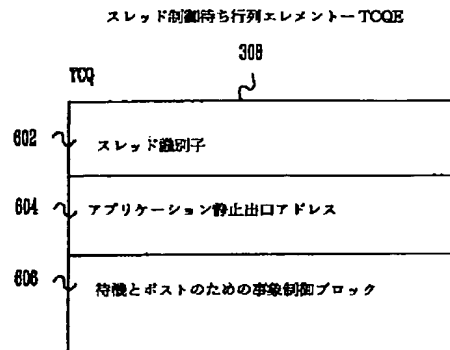
【図 4】



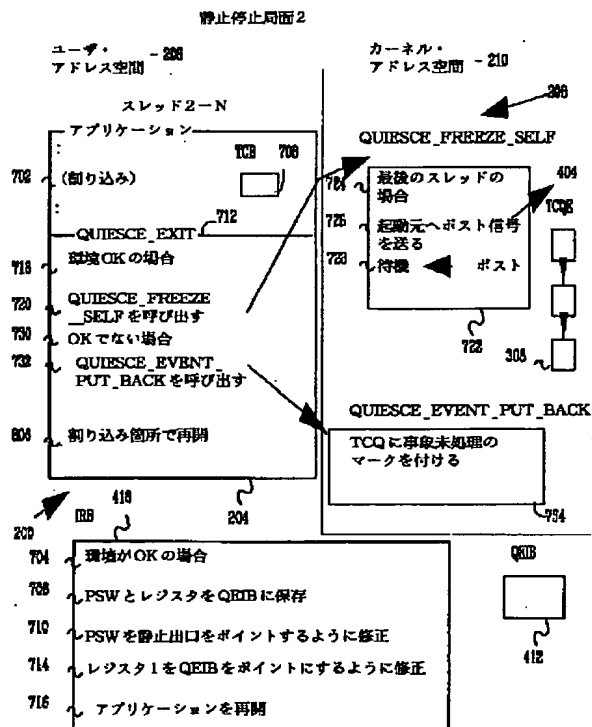
【図5】



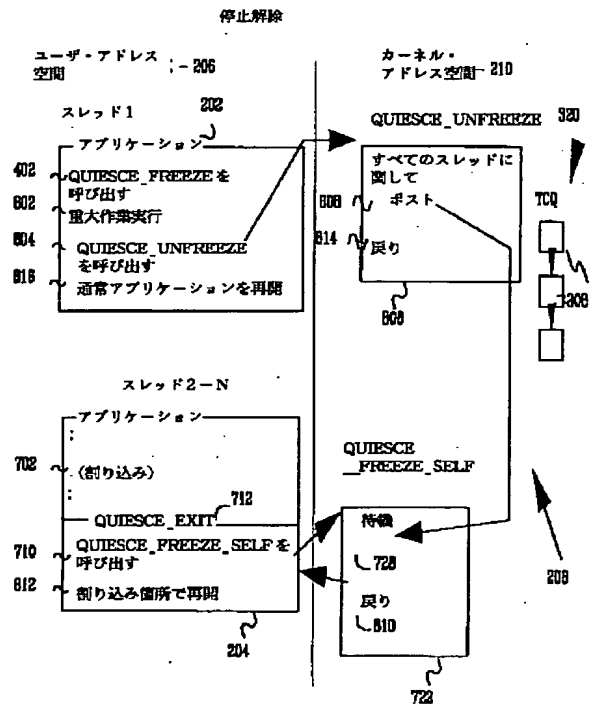
【図6】



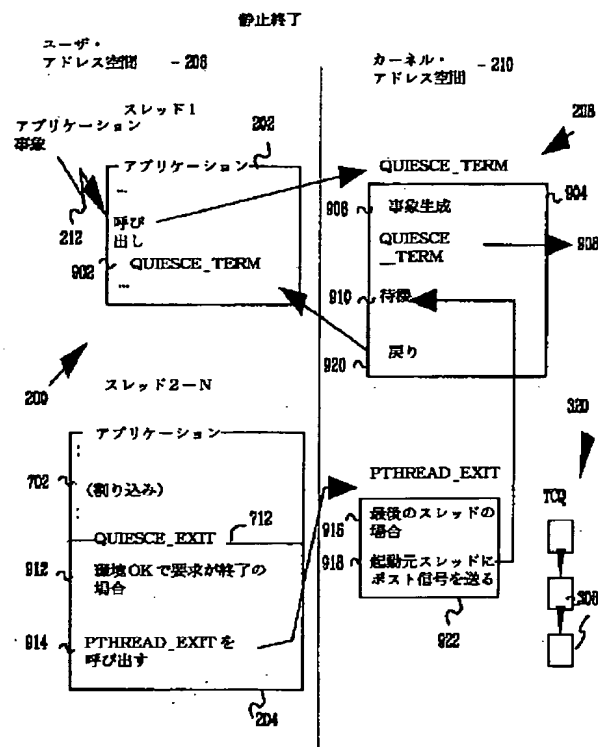
【図7】



【図8】



【図9】



フロントページの続き

(72) 発明者 アーネスト・スコット・ベンダー
アメリカ合衆国12477ニューヨーク州ソー
ガティース、バイン・グローブ・スクー
ル・ロード 27

(72) 発明者 ジョン・ケビン・フランク
アメリカ合衆国12477ニューヨーク州ソー
ガティース、デーブ・エリオット・ロード
186

(72) 発明者 ジョン・アーサー・ヘルムボルド
アメリカ合衆国12401ニューヨーク州キン
グストン、リンダーマン・アベニュー
583